User Password Authentication using Two Server Architecture

Soumya P. Bhandlkar[1] Minahaj N. Choudhari[2]
[1,2]Department of Computer Science
[1,2]VTU, Belagavi

Abstract— Nowadays the common methods used for user authentication are user id and password. In single server authentication system, information that is required to authenticate the user is stored in a single server. All the information is obtained when the server is compromised. In order to overcome this problem, a user password authentication using two server architecture is proposed where a secure password is divided and stored in two servers. If one server is compromised an attacker obtains only half of the authentication information. The user password is concatenated with the random string called salt and hash of salted password is then divided and stored in two servers. The two server's assistance is required to authenticate user and they run in parallel. In addition to providing the authentication, the proposed system also provides confidentiality by storing only half of the encrypted user files in each server.

Key words: Salting, Key Stretching, Iteration Count

I. INTRODUCTION

Nowadays password is used for identification and authentication of registered user, to an application during login process. The password provided by user, during registration need to be stored in the server in a secure manner. When the user enters the password during login process it should be compared with the stored password for authentication. Passwords are stored in the database and should not be stored in plaintext because when the server is hacked or compromised; all the passwords stored in the server are revealed [7]. So passwords should be stored in an encrypted form. This can be done in two ways:

- Reversible encryption functions such as DES, AES etc.
- Apply one way functions called hash functions such as MD5, SHA.

The right option for storing user's password is hashing because hashing is one way technique once hashed user cannot get his original input. Encryption is based on the key and is reversible; once anyone posses the key they can decrypt and obtain the original value. So it is cryptographically secure to store hash value of the password in the server.

Existing authentication systems which are password based transmits hash value over public channel. In today's computer era simply hashing the passwords and storing has many defects such as speed and recognizability [9].

1) Speed:

At a high speed guessing of password is done by hashing random strings and then comparing with the hash value which are stored in database [9]. By looking at the hash value it is not possible to deduce the password but if the password is not too long or complex it can be guessed easily. Most of the users chosen password is neither complex nor long [10].

2) Recognizability:

When the password is hashed it always returns the same hash value. There is a possibility that an attacker may spend time to compute hash of commonly used password and compare with the one stored in the accessed database. This form of attack is called as rainbow table attack. Rainbow table is a list of pre-computed hash of passwords which are commonly used [10], [12]. An attacker can attempt each entry of password and look-up hash value of password stored in rainbow table [9].

Furthermore, when an attacker access the hash value he can work offline and tries to guess password, computes hash and checks it with stolen hash. Dictionary attack and brute-force attacks are two common ways for guessing the password.

Typical authentication systems which are based on password used single server which stores information required to authenticate a user. All the information that is required to authenticate the user is disclosed, if that single server is compromised by an attacker. To overcome such issues two server authentication system based on password is used where mutual co-operation between two servers is required to authenticate the user. If one server is compromised means an attacker still cannot act like a user from the information obtained from the server [1]. The authentication information derived from password is stored in both the servers so that while authenticating a user both servers assistance is required.

A. Salting

Salting is one of the cryptographic techniques used in ciphering the password where a random string called salt is generated securely and concatenated to the password (password + salt) prior to hashing[11]. The salt is generated randomly each time when the user registers and added to the password. Salt is also used to prevent hash collision. There might be chances that two users having the same password, salt is used to differentiate these passwords, since each time salt is generated securely and randomly, the different results are obtained after hashing . The entropic nature of password may be weak; this weakness can be overcome by using salt. The strength of the password can be increased by adding salt to the password. If salt is of Sln bits then 2^{Sln} possible combinations can be done with the password. This makes the rainbow table attack hard [12]. Salt increases the difficulty of an attacker to find the user's having similar password. This solves recognizability problem. The Salting is used to overcome dictionary using brute force attack and rainbow table attack.

B. Key Stretching

Even if password is secured by hashing and using random salt, in this computer era any passwords can be cracked in more or less time by brute force attack using rainbow table and dictionary attack. To reduce the harm caused by the brute force attack algorithms which are CPU intensive such as PBKDF2 (password based key derivation function) is

used. PBKDF2 is one of the function for key derivation, used along with pseudorandom function such as HMAC for the salted password and the process is repeated many times to produce the salted PBKDF2 hash of the password. This is also called as key stretching. Its main idea is that once the hash function is applied to the password the same hash function is repeated till specified iteration count. The desired minimum iteration count is 1000 [8]. The result of PBKDF2 will be in the form of Hash(hash(hash(…….hash(password||salt)))). This function is not meant for hashing the passwords but its slowness property acts well for hashing the password. The output of this function is PBKDF2 hah of salted password which is iterated 1000 times [10].

The proposed system is symmetric where the two servers mutually cooperates for user authentication and is password based system [1]. In traditional two server's authentication system based on password, the user encrypts the password using the secret key and stored in servers [1], [2]. When this secret key is compromised means all the passwords stored in sever are obtained by the cracker. To overcome these issues the proposed system uses salt which is concatenated to the password and hashed. The beauty of the salt is that for each user chosen password, salt is generated randomly so that even though an attacker obtains the salt the attacker cannot obtain the password. In addition to dividing and storing the password authentication information in two servers the proposed system also divides user files and stored so that so that even if server is compromised means an attacker cannot obtain user's files.

II. Related Work

When single server system is used, all the information necessary to authenticate user are all disclosed. To overcome this issue Ford and Kaliski [7] proposed Thershold Pake protocol which is based on PKI model, where the client encrypts the password using server's public key and this model requires n server's to authenticate the client. The protocol is insecure when n-1 or few servers are disclosed.

MacKenzie et al. [3] introduced PKI based setting which suggested that out of n server's only s servers should co operate to authenticate the client. This protocol is insecure when s-1 or fewer servers are disclosed. They provided formal security proofs.

Brainard et al. [5] developed first PKI based model on two server which is implemented with the aid of public key techniques such as SSL which assumes secure channel between client and server.

Katz et al. [6] introduced password based key exchange protocol model on two servers which is built on Katz-Ostrovsky-Yang PAKE protocol(KOY protocol) where randomly chosen client password pw should be divided randomly for two servers A and B such that pw=pw1+pw2. This protocol executes twice. One between server A and client with the aid of server B to authenticate client and one between server B and client with the aid of server A. To authenticate the client other server help is required because the password is divided and stored and two servers. Each client and server agrees a secret session keys. Two servers assist each other to authenticate the client so Katz protocol is symmetric. This protocol is inefficient for practical use.

Yang et al. [4] introduced asymmetric PKI based two server PAKE protocol where the user interacts only with front end server called as service server while the back end server called control server helps service server to authenticate the client and the client and the service server agrees the secret session key. In 2006 yang and Jin came up with a password based two server PAKE protocols where the client sends request to service server. The service server responds $B=B_1B_2$ to client; where $B_1= g_1^{b1}g_2^{\pi 1}$ and $B_2= g_1^{b2}g_2^{\pi 2}$ are generated by control and service server, where π_1 and π_2 are the password of user shared randomly. After computing the client obtains g_1^{b1+b2} where $\pi = \pi_1 + \pi_2$. With the help of Control server, the client and service server agree secret session keys. Even though this protocol is more efficient from Katz's protocol and for practical use it is inefficient because the two servers need to compute in series and requires more computational rounds.

Jin et al [2] introduced two server PAKE protocol having less computational rounds than Yang's two server PAKE protocol. In this Jin's PAKE protocol the client sends $B= g_1^a g_2^{\pi}$ to the service server which in turn forwards $B1 = B/g_1^{b1}g_2^{\pi 1}$ to control server. The control server responds $A_1 = g_1^{b2}$ $B_2= (B_1/g_2^{\pi 2})^{b2}$ $=g_1^{(a-b1)\,b2}$ to service server then service server will compute $B_3= (B_2A_1^{b1})^{b3} =g^{ab2b3}$ and sends $A_2 = A_1^{b3}$, S1 =Hash(B_3) to the client. The client and service server authenticate each other and agrees to the same secret session keys. The disadvantage of this protocol is that two server need to compute in series.

III. Proposed System

The proposed system has 4 main phase's initialization, registration, authentication and providing service.

A. *Initialization*

- In the initialization phase the two servers need to be setup where the password authentication information and the user's files need to be stored.
- These two servers co operate each other in parallel to authenticate the user.

A. *Registration*

- In the registration phase the user should register to the servers by providing information such as userid, name, city, mobile number and the user must choose the password.
- The chosen password is sent to the web server interface.
- The web server interface generates random string called salt using secure random number generator and the salt is appended to the password.
- The web server interface then subjects the password to the PBKDF2 function which performs hash (hash (hash (…… (hash (password+salt))))) and derives the authentication information. It is impossible to derive the password in return from this authentication information.
- The iteration count, salt in plain text and the PBKDF2 hash of salted password, separated by colon is stored in the temporary file.

- The size of the temporary file is then divided by the total number of servers i.e. two.
- Now the system stores half authentication information in one server file and other half authentication information in second server file having userid as the file name.

B. Authentication

- When the user login to the system with the userid and password, the web server interface then performs same action over the password.
- It takes the files corresponding to userid name from the two servers which contain half authentication information of password.
- After combining both the files the web server interface obtains the salt stored in the file then subject's the password and salt to the PBDF2 function. It then compares both hashes if they are equal means the server will ensure that user is authentic, and then only the server will continue to provide the service.

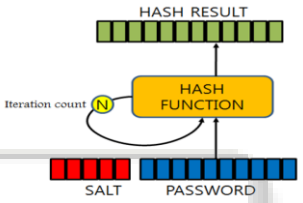

Fig 1: PBKDF2 hash of salted password.

C. Providing Service

- When the user is authentic the server will continue to provide the service.
- When the user uploads the file the web server interface encrypts the file using DES encryption. The user uses master key for encryption.
- The web server interface then splits the encrypted file based on the total number of servers i.e. two and then stores the half encrypted file in one server and another half encrypted file in second server such that even if one of the servers is hacked by an attacker he is not able to obtain the complete file of the authorized user. This overcomes single point of failure.
- In the single server system if the server is hacked means all the files are available to an unauthorized user. In the proposed system if one of the server is hacked means an attacker will obtain only half of file.
- When the user sends the request message for downloading the files, the web server interface obtains the files from both the servers.
- After combing the files the user uses the same key which he has used to encrypt the files and decrypts the file for his use.

IV. Conclusion

In this paper, we have developed User password authentication system using two server architecture where the authentication information of password is divided and stored in two servers rather than single server thus providing secure authentication system. In single server password based authentication system all the information necessary to authenticate the user are stored. If server is hacked all the information are disclosed. In order to overcome this problem, a user password authentication using two server architecture has been proposed where a secure password is divided and stored in two servers. If one server is compromised an attacker obtains only half of the authentication information. It also provides confidentiality of the user files by storing only half encrypted files in two servers.

V. Future Work

The proposed system uses only two server architecture. This can be extended to multiple servers to enhance security. The proposed system uses master key to encrypt the user files. In future work the user files can be encrypted using secret key of each user and then securing the key.

References

[1] Xun Yi, San Ling, and Huaxiong Wang, "Efficient Two-Server Password-Only Authenticated Key Exchang", IEEE Transactions on Parallel And Distributed Systems, September 2013.
[2] H. Jin, D.S. Wong and Y. Xu, "An Efficient Password-Only Two-Server Authenticated Key Exchange System", Proc. Ninth Intl Conf. Information and Comm. Security (ICICS 07), pp. 44- 56, 2007.
[3] J. Katz, P. MacKenzie, G. Taban, and V. Gligor, "Two-Server Password-Only Authenticated Key Exchange," Proc. Applied Cryptography and Network Security (ACNS '05), pp. 1-10, 2005.
[4] Y. Yang, F. Bao and R.H. Deng, "A New Architecture for Authentication and Key Exchange Using Password for Federated Enterprise", Proc. 20th IFIP Intl Information Security Conf. (SEC05), pp. 95-111, 2005.
[5] J. Brainard, A. Jueles, B.S. Kaliski and M. Szydlo, "A New Two- Server Approach for the Authentication with Short Secret," Proc. 12th Conf. USENIX Security Symp., pp. 201-28, 2003.
[6] J. Katz, R. Ostrovsky and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques: Advances in Cryptology (Eurocrypt '01), pp. 457-494, 2001.
[7] W. Ford and B.S. Kaliski Jr., Server-Assisted "Generation of a Strong Secret from a Password", Proc. IEEE Ninth Intl Workshop Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 110-180, 2000.
[8] Jerry Orr, "Secure Password Storage", http://www.Jerry on Java_ Secure Password Storage – Lots of don'ts, a few dos, and a concrete Java SE example.html, May 2012.
[9] Patrick Mylund Nielsen, "Storing Passwords Securely", http://storing password securely.html, June 2012.
[10] Veera Sundar, "Storing passwords in Java web application", http://storing passwordsin java web application javalobby.html. 207.
[11] "Salt(cryptography)",http://en.wikipedia.org/wiki/Salt_(cryptography).
[12] "RainbowTable",http://en.wikipedia.org/wiki/Rainbow_table.

